Cooperative Visualization of Computational Fluid Dynamics

Michael J. Gerald-Yamasaki

RNR Technical Report RNR-91-011, March, 1991

Numerical Aerodynamic Simulation Systems Division
NASA Ames Research Center, Mail Stop T045-1
Moffett Field, California 94035-1000
yamo@nas.nasa.gov

March 4, 1991

Cooperative Visualization of Computational Fluid Dynamics

Michael J. Gerald-Yamasaki

Numerical Aerodynamic Simulation Systems Division
NASA Ames Research Center, Mail Stop T045-1
Moffett Field, California 94035-1000
yamo@nas.nasa.gov

March 4, 1991

Abstract

Interview *is a computational fluid dynamics visualization application for which processing is distributed between high performance graphics workstations and supercomputers. Facilities are provided in the application for more than one user to view shared images creating a cooperative visualization environment. The way in which the computation is partitioned between the supercomputer and the workstations is critical to the capability of the application to present simultaneous, identical, animated images of fluid dynamics to more than one user.*

1. Introduction

Scientific visualization has become increasingly important in the analysis of large data sets. The pattern-recognition capabilities of the visual sense are utilized to analyze much greater quantities of visually transformed data than is possible with purely numeric data [8, 20].

Recently developed computer tools which facilitate the collaborative process have found that a WYSIWIS (what you see is what I see) interface is valuable. This type of interface provides for the "presentation of consistent images of shared information to all participants"[23]. Such an interface applied to visualization would allow scientists to see and interact with each other's work through their workstations.

For an application such as the visualization of computational fluid dynamics (CFD) [2], the technological requirements for a cooperative system can be daunting [13]. How the partitioning of the computation is accomplished will determine which visualization techniques are both interactive and cooperative.

The problem might be stated as how can one best utilize the capabilities of high performance graphics workstations, supercomputers and networks to provide a system for visualizing complex CFD data cooperatively.

2. Background

In CFD the increased capabilities of supercomputers has in turn dramatically increased the size and complexity of numerical simulations of fluid flow [19]. As the size of the simulations increases, the size of the solution data also increases and can result in data sets representing the physical characteristics of a flow field which are immense.

Despite advances in the delivery of computational power to users of high-performance graphics workstations, there remain visualization applications for which the computational requirements can only be met by supercomputers. The combined capabilites of the supercomputer and the high-performance graphics workstation can be utilized in distributed visualization applications. There are a number of different approaches to distributing the visualization process over supercomputers, graphics workstations and other machines [5, 17, 18, 19, 25].

Visualization techniques which utilize the advanced capabilities of high-performance graphics workstations are for the most part not transportable beyond the workstation without some loss of informational content: resolution degradation, color degradation, lack of motion or animation, or lack of interactivity. This lack of transportability makes it difficult to communicate the results of the visualization process to collaborators.

Interview is a tool for visualizing CFD. Processing in Interview is distributed between a high performance graphics workstation and a supercomputer. The computational environment on the supercomputer may be shared with another workstation to provide a cooperative visualization environment.

The development of Interview is necessarily an interdisciplinary endeavor: CFD, graphics, distributed processing, supercomputing, and computer-supported cooperative work (CSCW). This paper focuses on the CSCW and distributed processing aspects of Interview, discussing the other aspects as necessary to understand the overall application.

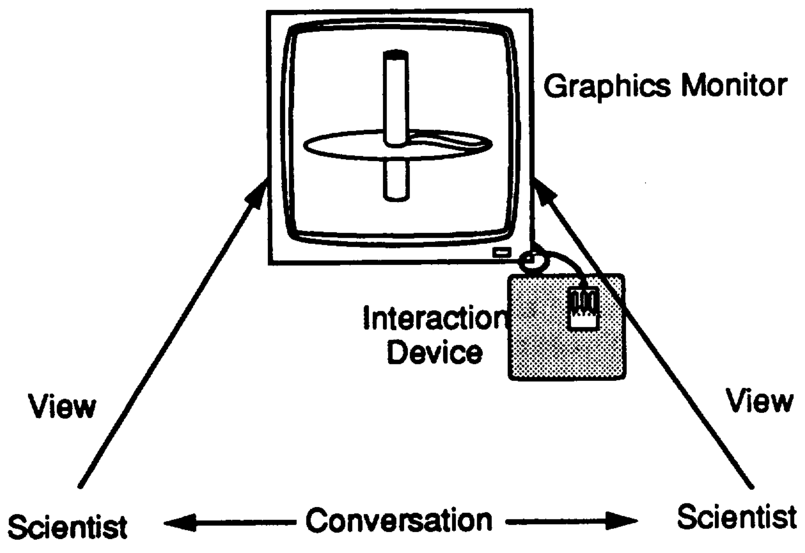
3. Facilitating Collaboration

One model of a collaborative environment is represented in this simple diagram [21]:

$$\begin{tikzcd}
& \text{Shared Space} \arrow[dl, leftrightarrow] \arrow[dr, leftrightarrow] & \\
\text{Receiver/Sender} & \text{Conversation} \arrow[l] \arrow[r] & \text{Receiver/Sender}
\end{tikzcd}$$

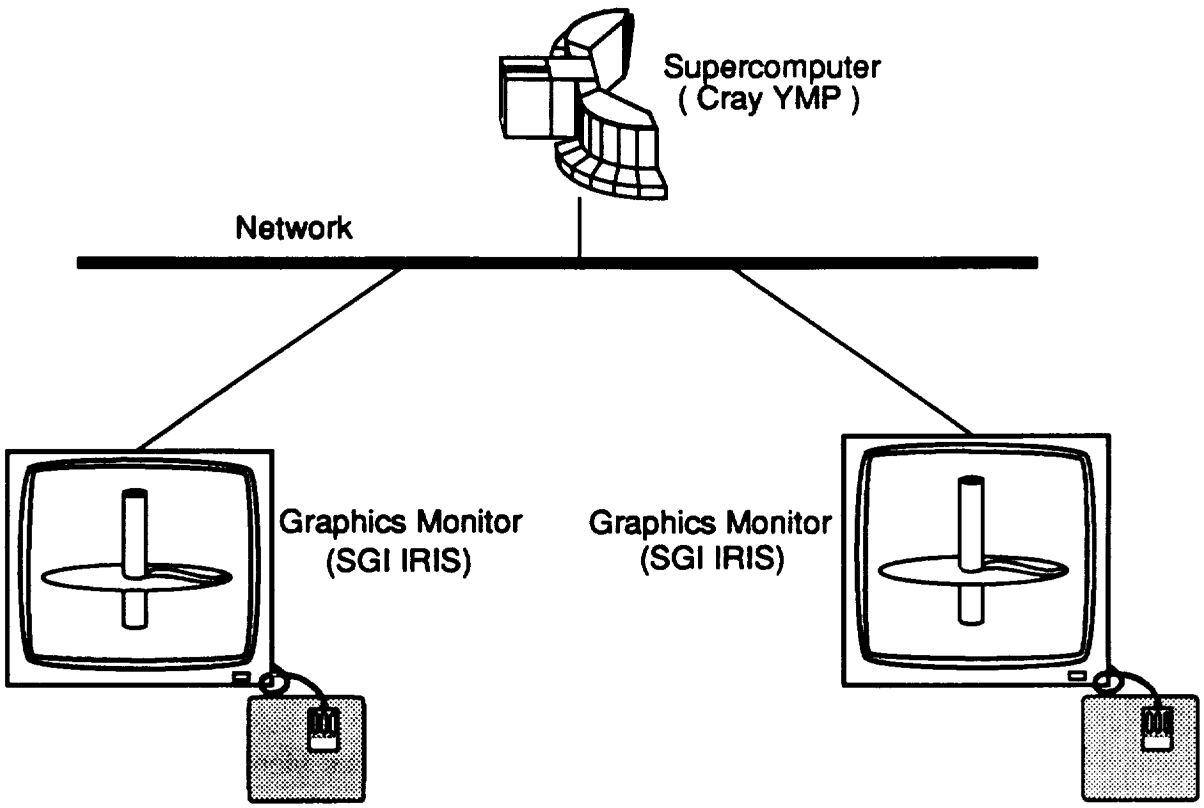
The idea of WYSIWIS follows somewhat from this basic model. The shared access to information makes the symbolic representation of the data more concrete.

In the typical visualization application a single user is "alone" with her data. When an interesting image is produced on the graphics monitor, it's common in our laboratory to call co-workers to the monitor to see what has been produced. With the shared view of the monitor, the collaborative environment outlined above is created.



Collaborators who are in another building or another city however, cannot participate in this environment.

Interview builds on the basic model to provide an environment where distance is not a deterrent to creating the collaborative environment. For Interview, the shared space resides on the supercomputer, the images are rendered on separate graphics workstations and can present either identical or different images.



4. The Computational Environment

One of the main objectives of the Numerical Aerodynamic Simulation (NAS) Program at NASA Ames Research Center is the provision of a comprehensive computing environment to facilitate computational aerodynamics and fluid dynamics research [1]. To this end the NAS Processing System Network (NPSN) was developed. The NPSN contains a wide range of computer systems, including two high-speed processors (currently, a Cray 2 4/256 and a Cray YMP 8/128) and a small army of Silicon Graphics IRIS workstations. Several networks are employed to provide connectivity and a basis for network development and research. These networks include Ethernet, HYPERchannel, UltraNet, and Pronet-80.

The main vehicle for distributing computation between supercomputers and workstations in Interview is Distributed Library (dlib) [27]. Like many systems which provide for distributed processing, dlib is based on the remote procedure call (RPC) model [3, 9, 24, 26]. However, unlike most of these systems, dlib was developed to provide a service which allows for a conversation of arbitrary length within a single context between client and server. The dlib server process is designed to be capable of storing state information which persists from call to call, as well as allocating memory for data storage and manipulation. While RPC protocols are frequently likened to local procedure calls without side effects, dlib more closely resembles the extension of the process environment to include the server process.

5. The CFD Application

The process involved in CFD research can be broken down into three steps: grid generation, numerical simulation of fluid flow, and post-processing of the resulting flow solution data. A numerical grid is created describing an object and the fluid space surrounding the object. Flow solvers calculate physical properties of the flow at the nodes of the grid. The flow solution can be steady state, in which the physical properties at each node do not change over time, or unsteady, in which changes in the physical properties are observed over time.

Typical post-processing data sets consist of a grid file, containing the x, y, and z coordinate values for corresponding grid nodes, and a solution file, containing the values for density, energy, and momentum in three dimensions for each grid node. Density and energy are scalar values while momentum is a three dimensional vector. A steady state solution data set would contain a grid file and a solution file. An unsteady solution data set would contain a grid file and a solution file per time step. Typical grid sizes can be as large as several million nodes. Due to storage considerations unsteady solution data sets are usually truncated in some manner but can still consume multiple gigabytes of storage space.

With the basic solution values of density, energy and momentum, other physical characteristics of the flow can be calculated. Other scalar values that may be calculated include such quantities as temperature, pressure, velocity magnitude, and kinetic energy. Other vector fields include velocity, vorticity, and gradients of scalar fields such as the pressure gradient. A variety of visualization techniques can be applied to these scalar and vector fields.

To get an idea of how the visualization process works in a distributed and cooperative environment, the data of the numerical simulation of the unsteady fluid flow past a tapered cylinder [14] will be used as an example. The grid for the tapered cylinder is relatively small at 128 K nodes. A data set representing 256 time steps is used for the visualization.

Tapered Cylinder Data:

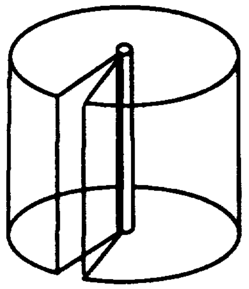
Grid: 32 x 64 x 64 nodes, 131,072 nodes, or 393,216 words

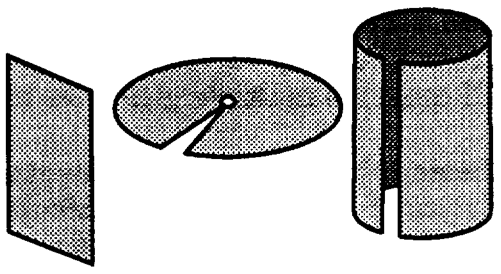
IRIS format:	1,572,864 bytes
Cray format:	3,145,728 bytes

Solution: 256 time steps for 5 physical values, 167,772,160 words

IRIS format:	671,088,640 bytes
Cray format:	1,342,177,280 bytes

There are a variety of visualization techniques which can be applied to this data but this discussion will be limited to grid surfaces colored by a scalar value. A grid surface is a 2-D subset of the 3-D grid.


Wire frame of full tapered cylinder grid


Examples of grid surfaces
Approximate data size over 256 time steps: 4 Mbytes

6. Problem Decomposition

Again, the problem might be stated as how can one best utilize the capabilities of high performance graphics workstations, supercomputers and networks to provide a system for visualizing complex CFD data cooperatively.

One method for visualizing a data set like the tapered cylinder is to look at a physical characteristic of the flow over time. Calculated values for, say, velocity magnitude can be mapped to color values and displayed for a grid surface. The area between the grid points can be rendered with interpolated color values using the Gouraud shading capability of the IRIS Graphics Library. Successive time steps can be rendered to create an animation.

The basic steps involved in creating a grid surface animation are:

Step 1. Extract data for grid surface from solution data set.

Step 2. Calculate scalar values for each node in the grid surface data.

Step 3. Map scalar values to color values.

Step 4. Render and display surface with Gouraud shaded polygons.

Step 5. Animate by displaying surfaces sequentially by time step.

Since the environment has many tools available, "using the appropriate tool for the task" should be one guide. In order to use the appropriate tool, one must understand their capabilities. Simply stated:

The supercomputer -

- large number of MFLOPS
- large memory space
- large disk space
- fast disk to memory I/O
- fast I/O to networks
- 64 bit word

The workstation -

- high performance graphics
- user interface tools
- medium size memory
- medium size disk
- relatively slow disk speed
- medium I/O speed to networks
- 32 bit word

Distributing the tasks between the workstation and the supercomputer results in this organization:

i. Store original flow solution and grid data on Cray disk in IRIS format. This puts the smallest format for the data on the fastest medium for the data.

ii. Read grid data into Cray memory. The grid data may remain in IRIS format as, for this visualization technique, no numerical manipulation of the grid is required.

iii. Extract solution data for grid surface from disk. To optimize reads from disk the flow data has been organized to contain the values for the five physical characteristics over all the time steps for each node in contiguous memory space on disk. This allows for reads in larger blocks.

iv. Translate extracted data to Cray format for numerical manipulation. Translation routines are fully vectorized and very fast.

v. Calculate the scalar value over the grid surface data for all the time steps. Calculation over large amounts of data is the best utilization of Cray CPU power.

vi. Map scalar values to color values, again using Cray CPU power.

vii. Translate color values to IRIS format.

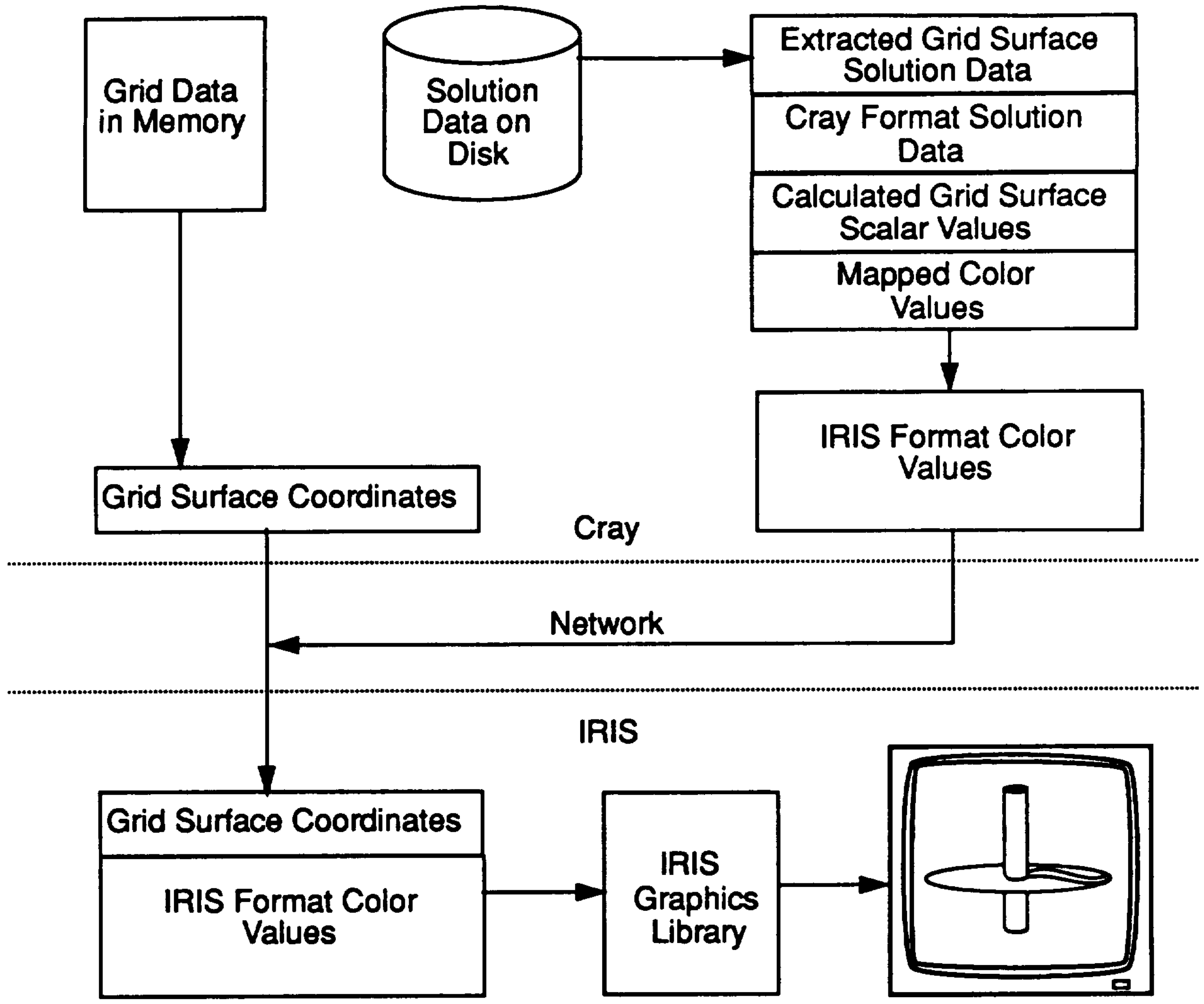
viii. Extract grid coordinate data for grid surface from grid data in Cray memory (IRIS format).

ix. Transfer grid coordinate data and grid surface color values to IRIS over network.

x. Use grid coordinate information as vertices for IRIS Graphics Library calls with grid surface color values. Gouraud shade polygons formed by vertices to interpolate color over the surface.

xi. Display sequentially one time step at a time to animate grid surface.

The following is a diagram of the process outlined above.



Each of the steps which are performed on the Cray are implemented with dlib calls with the Cray acting as the server machine. Using dlib requires an initial call to dl_init, which establishes a connection between the client and an application-specific server. The server is a process on the remote machine which behaves as an extension of the application's environment. The remote process persists from dlib call to dlib call and state information, such as open file descriptors or allocated memory, as part of this process also persist. The remote process receives dlib calls from the client via the network, executes the designated subroutine, and sends return values back to the client over the network.

Step iii above is an example of how dlib is used and is described in detail below. Dlib calls are invoked on the IRIS, so here "local" refers to the IRIS and "remote" refers to the Cray.

1. Memory to contain the grid surface data is allocated on the Cray using

the standard dl_malloc dlib call. This returns a memory descriptor for the remote memory segment.

2. The solution file on the Cray is opened with the standard dl_open call.

3. An application-specific dlib routine is implemented which performs the sequence of reads and seeks to extract the solution data for the desired grid surface from the remote solution file. The memory descriptor from step 1 is passed as an argument to this routine as the buffer for the extracted data.

4. The result is a buffer on the Cray which contains the solution data for the desired grid surface which can be referred to by a memory descriptor.

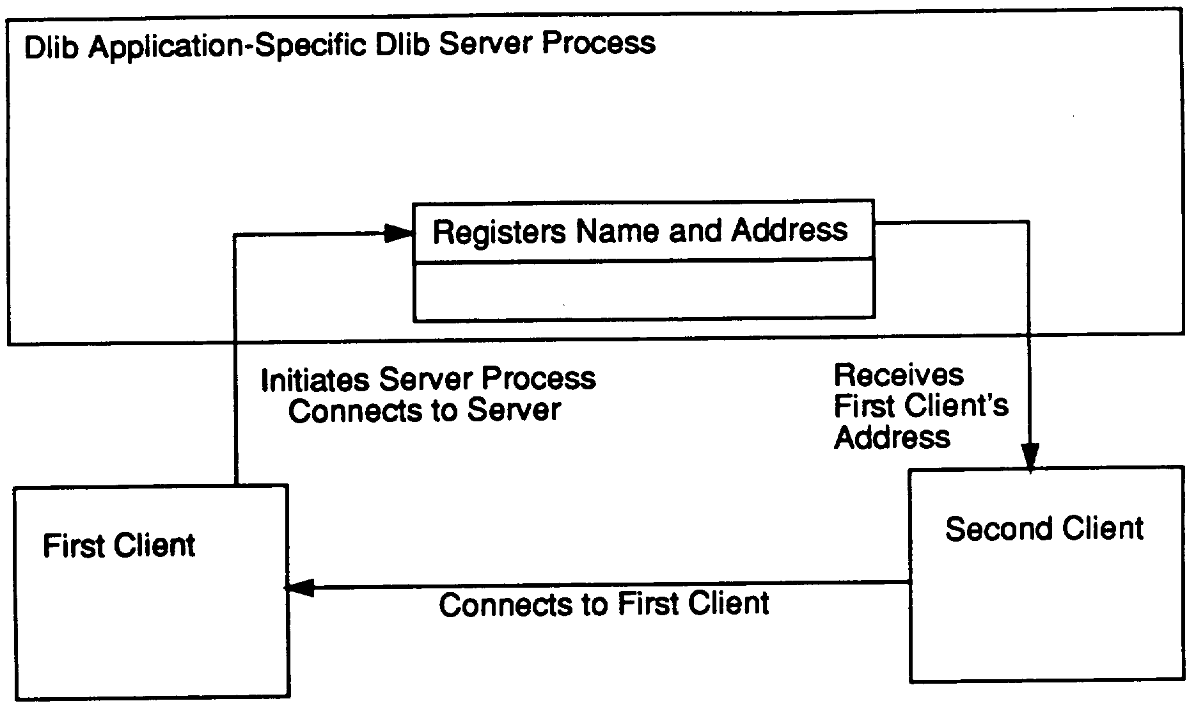
Memory descriptors remain valid until the memory is explicitly freed using dl_free. The memory descriptor in the example above would be further used (step iv) as an argument to a dlib data translation routine.

7. WYSIWIS

The dlib application-specific server provides a substrate for the shared space of the collaborative environment model. The CFD data to be visualized, and the intermediate results of graphics processing (e.g., mapped color values, coordinate information), are contained in the server process.

Dlib was originally designed on a model of one client to one server. To allow multiple clients to share the server process environment, the dlib server was modified to accept more than one connection. Each connection is selected for service by the server process as dlib calls are received.

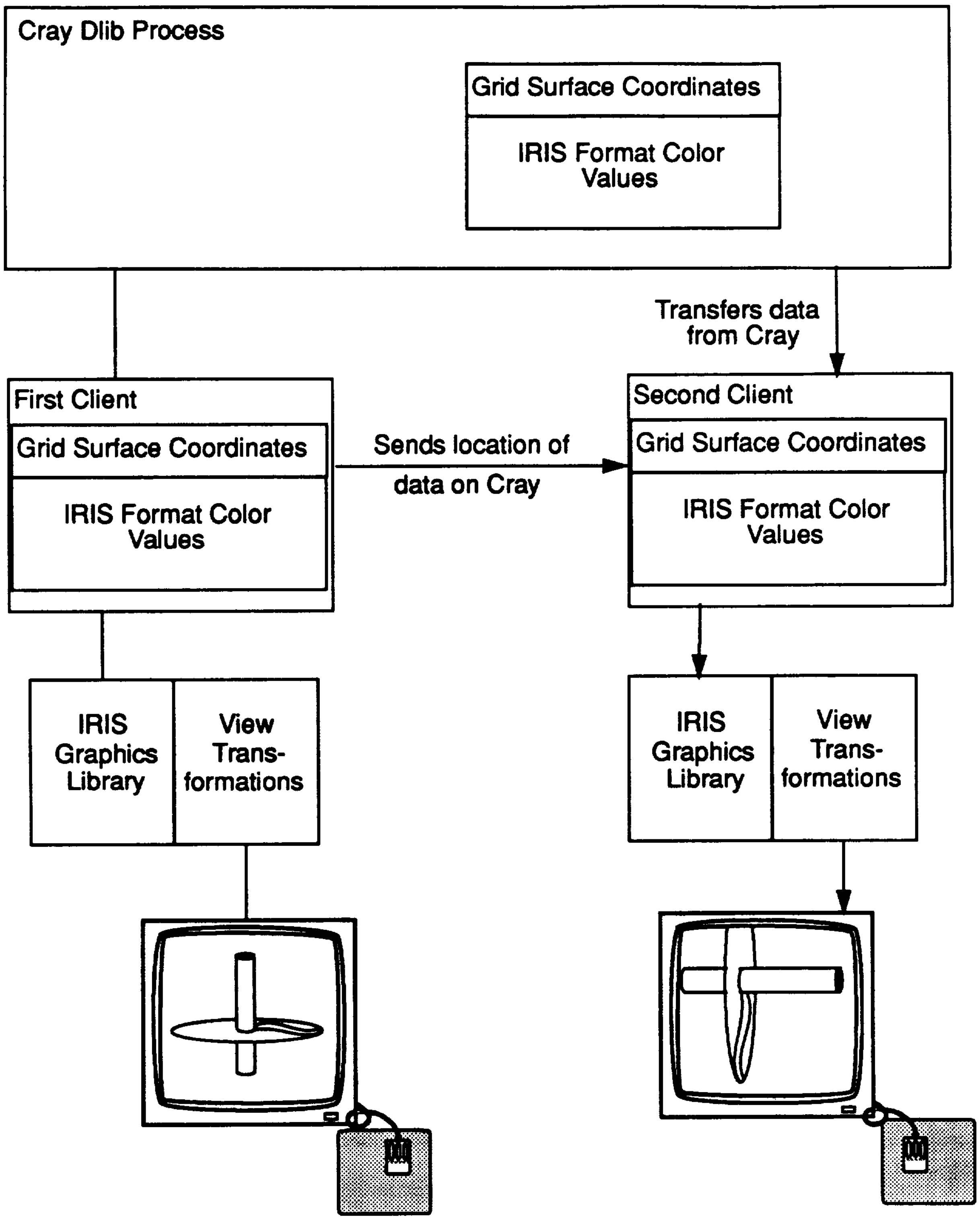
In Interview three connections are created: first client to server, second client to server, and second client to first client. The first client initiates the dlib server, connects to the server, and registers in the server an address at which the first client is accepting connections. The second client connects to the server, reads the first client's address, and connects to the first client.



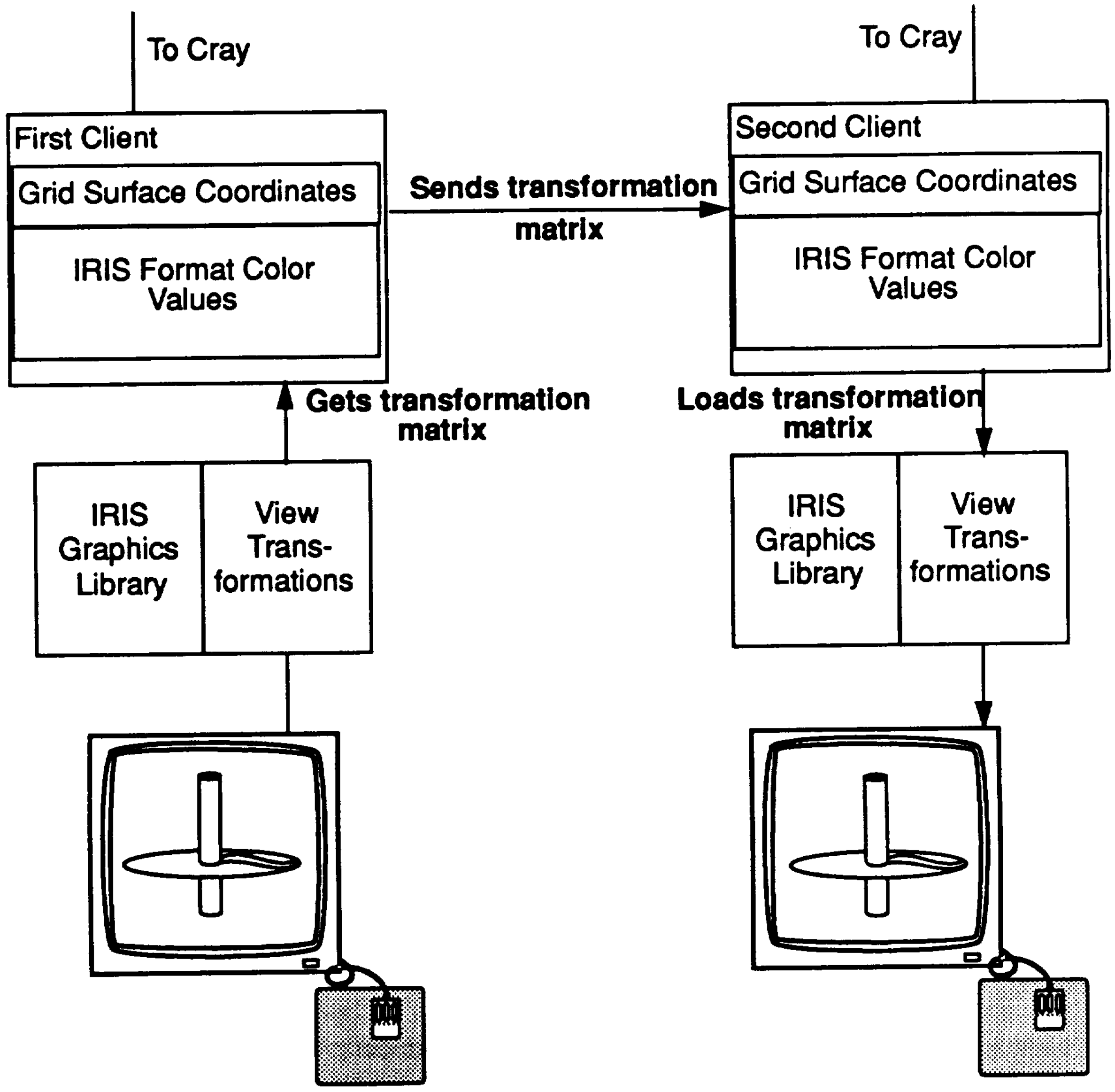
The first client will have been active for an arbitrary length of time when the second client is invoked. As such, the image data the first client is presenting may contain a number of grid surfaces. The first client maintains a list of descriptions of the grid surfaces it is currently viewing. Upon request, this list is sent to the second client. From this list, the second client has the information to allow it to transfer the image data from the server process using dlib calls in the same way that the first client received this data.

The second client now is able to view the same image data as the first client with its own view transformation and animation sequencing. Interview provides mouse driven interaction to zoom, translate, and rotate the 3-D images. Mouse driven interaction is also used to control the animation sequencing. These controls are individual to each client. Consequently, the two clients at this point are viewing the same 3-D image data but may have different viewing perspectives.

The ability to individually view the same data is analogous to two people looking at a 3D object, say, an open book. One person can see the title and author on the front cover, while the other can read the pages. While their views are different, there is a shared context for conversing.

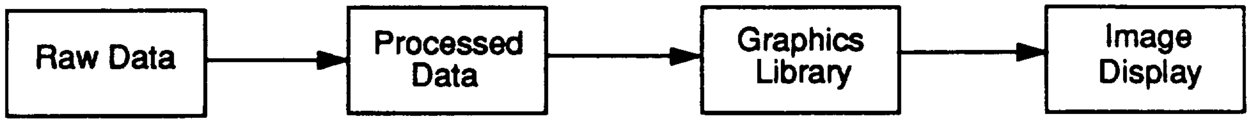

Cray Dlib Process
Grid Surface Coordinates
IRIS Format Color Values
Transfers data from Cray
First Client
Grid Surface Coordinates
IRIS Format Color Values
Sends location of data on Cray
Second Client
Grid Surface Coordinates
IRIS Format Color Values
IRIS Graphics Library
View Trans- formations
IRIS Graphics Library
View Trans- formations

To present a consistent image on both monitors simultaneously view transformations and animation sequencing must be consistent. The transformation matrix controls the mapping between coordinate data and the screen representation. In order for consistent images to be viewed by both clients, the first client gets a copy of its current transformation matrix and sends it to the second client. The second client in turn loads this new transformation matrix, creating a consistent image on both monitors. Animation sequencing information can also be exchanged to synchronize the animation frame by frame.



8. Discussion

There are many ways of distributing the computational workload for graphics applications between supercomputers and workstations [7, 11, 15, 18]. The impact of the network on the performance of these systems is certainly dependent on the bandwidth and latency of the network, but also the performance is dependent on the partitioning of the computation. The following is a simple block diagram of a visualization process:



One way to partition the computation of this process is to complete everything except the image display on the supercomputer [12, 13, 15, 16, 17, 18]. Image data is then transferred over the network to a workstation or frame buffer for display. This can turn out to be quite a large amount of data to transfer over the network; perhaps, 24 bits per pixel by 1024 by 1280 pixels per frame. The data transfer rate at 24 frames per second would then be approximately 90 MBytes per second. Maintaining this transfer rate to two or more workstations or frame buffers in a cooperative effort would be difficult.

While the advent of high speed networks in the gigabit-per-second range [4, 6, 10] removes a potential bottleneck in the performance of these distributed applications, high speed networks are not a panacea. Even though some networks may be capable of transferring data at a rate of a gigabit-per-second, it will be some time before workstations are capable of transferring or receiving data at that speed. Even so, image data transfer at animation speeds would be difficult to sustain.

A second way to partition the distributed visualization problem is to use the graphics library calls as the medium of exchange between the client and server [5, 18, 22]. This method is in essence a remote procedure call package for graphics routines. While requiring less data to be transferred than the image transfer method, a difficulty arises for interactivity. For real-time graphics manipulations such as handling mouse interactions, the network transactions required to handle the interactions can greatly reduce response time.

Interview places the partition between the processed data and the graphics library calls. In this way the minimum amount of data is transferred over the network. Interactivity and animation control is handled locally without network overhead. Only when identical views are to be shared by cooperating processes are network transactions required to handle interactivity. With the processing partitioned in this way, Interview is operational over networks with modest transfer rates, such as

Ethernet. Network speed has its greatest impact on the transfer of the processed data from the supercomputer to the workstation, but does not impact the viewing of animations. For transmission of the transformation matrices, the network only need transmit between 20 and 30 small packets per second to support full animation speeds. This is well within Ethernet's range.

9. Conclusion

The visualization of CFD is in essence an attempt to understand the physical characteristics of simulated fluid flow by examining subsets of a large data set represented by visual symbols, colors, and shapes. For unsteady flow solutions, animation adds another dimension to the analysis. It is difficult to share the process of analysis without a means to share the visual images. Not just viewing the visual images, but the whole process of interacting with the visual images, of selecting subsets, of controlling the animation, are important elements of collaboration. Interview demonstrates that careful partitioning of processing between supercomputers and workstations can produce an efficient system for cooperative visualization of CFD.

10. Acknowledgments

The author would like to thank E. Lisette Gerald, Jeff Hultquist, and Sam Uselton for reviewing early versions of this manuscript and suggesting numerous improvements to the final presentation.

11. References

[1] Bailey, F. R. Status and projections of the NAS program. In *Computational Mechanics - Advances and Trends.* A. K. Noor, editor New York: American Society of Mechanical Engineers, 1986, 7-21.

[2] Bancroft, G., Plessel, T., Merritt, F., Walatka, P. P., and Watson, V. Scientific visualization in computational aerodynamics at NASA Ames Research Center. *Computer* 22, 8 (Aug. 1989) 89-95.

[3] Birrell, A. D., and Nelson, B. J. Implementing remote procedure calls. *ACM Trans. on Comp. Sys.* 2, 1 (Jan. 1984), 39-59.

[4] Chlamtac, I. and Franta, W. R. Rationale, directions, and issues surrounding high speed networks. 78, 1 (Jan. 1990), 94-120.

[5] Choi, D. and Levit, C. Implementation of a distributed graphics system. *Internat. J. Supercomput. Appl.* (Winter 1987), 82-95.

[6] Clinger, M. Very high speed network prototype development: Measurement of effective transfer rates. In a report to NASA Ames Research Center in satisfaction of Contract #NAS2-12332/CTO#9, (Oct. 1989).

[7] Cook, L., Bancroft, G., Hussey, K., Dragon, J., and Johnston, W. Hardware/software solutions for scientific visualization at large scientific research laboratories. Panel proceedings of SIGGRAPH '89 (Boston, Mass., Jul. 31-Aug. 4, 1989) In *Computer Graphics* 23, 5 (Dec. 1989), 137-157.

[8] DeFanti, T. A., Brown, M. D., and McCormick, B. H. Visualization - Expanding scientific and engineering research opportunities. *Computer* 22, 8 (Aug. 1989), 12-25.

[9] Dineen, T. H., Leach, P. J., Mishkin, N. W., Pato, J. N., and Wyatt, G. L. The network computing architecture and system: an environment for developing distributed applications. *Proceedings of Summer Usenix* (June 1987), 385-398.

[10] Farber, D. Gigabit network testbeds. *Computer* 23, 9 (Sep. 1990), 77-79.

[11] Fowler, Jr., J. D., and McGowen, M. Design and implementation of a supercomputer frame buffer system. In *Proc. Supercomputing '88* (Orlando, Florida, Nov. 14-18, 1988) Washington, DC: IEEE Computer Society Press (Order No. 882), 140-147.

[12] Haber, R. B. Scientific visualization and the rivers project at the National Center for Supercomputer Applications. *Computer* 22, 8 (Aug. 1989), 84-89.

[13] Haber, R. B., and McNabb, D. A. Eliminating distance in scientific computing: an experiment in televisualization. *Internat. J. Supercomput. Appl.* 4, 4, (Winter, 1990), 71-89.

[14] Jespersen, D. and Levit, C. Numerical simulation of flow past a tapered cylinder. *NAS Applied Research Technical Report* RNR-90-021 (Oct. 1990).

[15] Johnston, W. E., Hall, D. E., Huang, J., Rible, M., and Robertson, D. Distributed scientific video movie making. In *Proc. Supercomputing '88* (Orlando, Florida, Nov. 14-18, 1988) Washington, DC: IEEE Computer Society Press (Order No. 882), 156-162.

[16] Phillips, R. L. A scientific visualization workbench. In *Proc. Supercomputing '88* (Orlando, Florida, Nov. 14-18, 1988) Washington, DC: IEEE Computer Society Press (Order No. 882), 148-155.

[17] Phillips, R. L. Distributed visualization at Los Alamos National Laboratory. *Computer* 22, 8 (Aug. 1989), 70-77.

[18] Phillips, D., Pique, M., Moler, C., Torborg, J., and Greenberg, D. Distributed graphics: Where to draw the lines? Panel proceedings of SIGGRAPH '89 (Boston, Mass., Jul. 31- Aug. 4, 1989) In *Computer Graphics* 23, 5 (Dec. 1989), 257-280.

[19] Rogers, S. E., Buning, P. G., and Merrit, F. J. Distributed interactive graphics applications in computational fluid dynamics. *Internat. J. Supercomput. Appl.* 1, 4 (Winter 1987), 96-105.

[20] Salzman, D. Visualization in scientific computing: Summary of an NSF-sponsored panel report on graphics, image processing, and workstations. *Internat. J. Supercomput. Appl.* 1, 4 (Winter 1987), 106-108.

[21] Schrage, M. *Shared Minds.* New York: Random House, 1990

[22] Silicon Graphics Computer Systems. Using the distributed graphics library. *4-Sight Programmer's Guide* Document Number 007-2001-030 (1990).

[23] Stefik, M., Foster, G., Bobrow, D. G., Kahn, K., Lanning, S., and Suchman, L. Beyond the chalkboard: Computer support for collaboration and problem solving in meetings. *Commun. ACM* 30, 1 (Jan. 1987), 32-47.

[24] Sun Microsystems. *Request for Comment #1057* Network Working Group (June, 1988).

[25] Treinish, L. A., Foley, J. D., Campbell, W. J., Haber, R. B., and Gurwitz, R. F.. Effective software systems for scientific data visualization. Panel proceedings of SIGGRAPH '89 (Boston, Mass., Jul. 31- Aug. 4, 1989) In *Computer Graphics* 23, 5 (Dec. 1989), 111-136.

[26] Xerox Corporation. Courier: the remote procedure call protocol. *Xerox System Integration Standard (XSIS) 038112*, (Dec. 1981).

[27] Yamasaki, M. Distributed library. *NAS Applied Research Technical Report* RNR-90-008 (Apr. 1990).